Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №3

«Основы теории чисел и их использование в криптографии»

Студентка: Пунько А.А,

ФИТ 3 курс 5 группа

Преподаватель: Берников В. О.

Минск 2020

**Цель:** приобретение практических навыков выполнения операций с числами для решения задач в области криптографии и разработка приложений для автоматизации этих операций.

**Задачи:**

1. Закрепить теоретические знания по высшей арифметике.
2. Научиться практически решать задачи с использованием простых и взаимно простых чисел, вычислений по правилам модулярной арифметики и нахождению обратных чисел по модулю.
3. Ознакомиться с особенностями реализации готового программного средства L\_PROST и особенностями выполнения с его помощью операций над простыми числами.
4. Разработать приложение для реализации указанных преподавателем операций с числами.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

В основе современной криптографии лежит теория чисел. Теория чисел или высшая арифметика – раздел математики, изучающий натуральные числа и иные похожие величины. В зависимости от используемых методов в теории чисел рассматривают несколько направлений. Нас будут интересовать вопросы делимости целых чисел, вычисления наибольшего общего делителя (НОД), разложение числа на простые множители, малая теорема Ферма́, теорема Эйлера, элементы теории вычетов.

Множество всех целых чисел (обозначим буквой Z) есть набор всех действительных чисел без дробной части: {..., -3, -2, -1, 0, 1, 2, 3, ...}. Натуральные числа являются подмножеством целых чисел и образуют множество N: {1, 2, 3, ...}. Делимость – одно из основных понятий теории чисел.

Если для некоторого целого числа a и натурального числа b существует целое число q, такое, что b\*q=a, то говорят, что число a делится на b. В этом случае b называется делителем числа a, а a называется кратным числа b. При этом используются следующие обозначения: a⋮b – a делится на b или b|a – b делит a. Из последнего определения следует, что:

* любое натуральное число является делителем нуля;
* единица является делителем любого целого числа;
* любое натуральное число является делителем самого себя.

Делитель a называется собственным делителем числа b, если 1<|a|<|b|, и несобственным – в противном случае.

Положительный наименьший собственный делитель составного числа n не превосходит √n. Всякое целое число а можно представить с помощью положительного целого числа b равенством вида а = bq + r, 0 ≤ r ≤ b. Число q называется неполным частным, а число r – остатком от деления а на b.

Каждое натуральное число, большее единицы, делится, по крайней мере, на два числа: на 1 и на само себя. Если число не имеет делителей, кроме самого себя и единицы, то оно называется простым, а если у числа есть еще делители, то составным.

Натуральное число n называется простым, если n > 1 и не имеет положительных делителей, отличных от 1 и n. Простое число не делится без остатка ни на одно другое число.

Любое составное число представляется уникальным образом в виде произведения простых чисел; иначе еще говорят, что разложение числа на простые множители однозначно. Это свойство вытекает из основной теоремы арифметики. Всякое натуральное число n, кроме 1, можно представить как произведение простых множителей: n = p1 · p2 · p3 · ... · pz, z > 1.

Для того, чтобы представить относительно небольшое число в виде простых сомножителей, достаточно уметь делить числа столбиком. Однако при этом следует придерживаться некоторых простых правил. Для первого деления нужно выбрать наименьшее простое число большее 1, которое делит исходное число без остатка. Частное от первого деления также нужно разделить с учетом указанных ограничений. Процесс деления продолжаем до тех пор, пока частным не будет 1.

Простых чисел бесконечно много, причем существует примерно n/ln(n) простых чисел, меньших числа n. Наименьший простой делитель составного числа n не превышает √n, поэтому для проверки простоты числа достаточно проверить его делимость на 2 и на все нечетные (а еще лучше простые) числа, не превосходящие √n; как видим, данное свойство коррелирует со свойством 1 собственного делителя. Из соотношения n=q\*p натуральных чисел, больших единицы, следует, что либо p, либо q принадлежит отрезку от 2 до √n. Поиск сомножителей числа n может вестись, например, перебором всех простых чисел до √n. Однако, если множители – большие простые числа, то на их поиск может потребоваться много времени.

Сложность решения задачи разложения больших чисел на простые сомножители, известной как проблема факторизации, определяет криптостойкость некоторых алгоритмов асимметричной криптографии, в частности алгоритма RSA. Любое четное число, большее 2, представимо в виде суммы двух простых чисел, а любое нечетное, большее чем 5, представимо в виде суммы трех простых чисел. Для любого натурального n, большего 1, существует хотя бы одно простое число на интервале от n до 2n.

Натуральное число n называется составным, если n > 1 и имеет, по крайней мере, один положительный делитель, отличный от 1 и n. Единица не считается ни простым числом, ни составным. Пример 6. Числа 17, 31 – простые, а числа 14, 15 — составные (14 делится на 2 и на 7, 15 делится на 3 и на 5). Вернемся к собственному делителю. Свойство 2 собственного делителя. Положительный наименьший собственный делитель составного числа n есть простое число. Так как простое число не делится ни на какое другое, кроме себя самого, очевидный способ проверки числа n на простоту – разделить n на все числа n1 и проанализировать наличие остатка от деления. Этот способ «в лоб» часто реализуется в компьютерных программах. Однако перебор может оказаться достаточно трудоемким, если на простоту нужно проверить число с количеством цифр в несколько десятков.

Понятно, что в криптографии используются числа, проверка на простоту которых производится гораздо дольше, и для работы с этими числами требуются специальные программные средства. К вопросу проверки чисел на простоту мы еще вернемся. Здесь же отметим, что первый алгоритм нахождения простых чисел, не превышающих n, был придуман Эратосфеном во 2 в. до н. э. и известен сейчас как «решето Эратосфена». Его суть в последовательном исключении из списка целых чисел от 1 до n чисел (или из сокращенного диапазона, например, от m до n, 1<m≤n), кратных 2, 3, 5 и другим простым числам, уже найденным «решетом». Как видим, описанное выше свойство 2 простых чисел и положено в основу рассматриваемого алгоритма. Для нахождения всех простых чисел не больше заданного числа n в соответствии с «решетом Эратосфена» нужно выполнить следующие шаги:

1. Выписать подряд все целые числа от двух (либо от m) до n (2, 3, 4, …, n). Пусть некоторая переменная (положим s) изначально равна 2 – первому простому числу.

2. Удалить из списка числа от 2s до n, считая шагами по s (это будут числа кратные s: 2s, 3s, 4s, …).

3. Найти первое из оставшихся чисел в списке, большее чем s, и присвоить значению переменной s это число.

4. Повторять шаги 2 и 3, пока возможно. Помимо рассмотренного алгоритма Эратосфена, который, понятно, является наименее эффективным, в настоящее время разработаны и используются другие алгоритмы.

Понятие делимости чисел является одним из важных в теории чисел. С этим понятием, а также с его производным – общим делителем связаны другие важнейшие (в частности, для криптографии) понятия: наибольшего общего делителя (НОД) и взаимно простых чисел.

Наибольшее целое число, которое делит без остатка числа a и b называется наибольшим общим делителем этих чисел, НОД (a, b). Понятно, что значение НОД можно вычислять для неограниченного ряда чисел. Простым и эффективным средством вычисления НОД (a, b) является метод или алгоритм ЕвклидаВ соответствии с этим определением используется цепочка вычислений двумя исходными (начальными) числами: а и b: аi = biqi + ri, 0 ≤ r*i* ≤ bi При i = аi и bi соответствуют как раз числам а и b. Последний ненулевой остаток (ri, i ≥ 0) соответствует НОД (a, b). Чтобы найти НОД нескольких чисел (например, a, b, c), достаточно найти НОД двух чисел (например, НОД(a, b) = d) потом НОД полученного (НОД(a, b)) и следующего числа (НОД(c, d) и т. д. Таким образом, чтобы вычислить НОД k чисел, нужно последовательно вычислить (k–1) НОД. Последнее вычисление дает искомый результат.

Взаимно простыми являются целые числа, наибольший общий делитель которых равен 1. Целые числа a и b взаимно просты тогда и только тогда, когда существуют такие целые u и v, что выполняется равенство: а\*u + b\*v = 1. Если НОД (a, b) = d , то справедливо следующее соотношение (соотношение Безу): аu + bv = d. Эта формула называется также реализацией «расширенного алгоритма Евклида». Этот алгоритм состоит из двух этапов: собственно алгоритма Евклида и вычислений на основе обратных подстановок или последовательного выражения остатков в каждом из шагов предыдущего этапа с соответствующим приведением подобных на каждом шаге.

Исследованием целых чисел занимался швейцарский математик Леонард Эйлер (Leonard Euler). Один из важных вопросов его исследования: сколько существует натуральных чисел, не превосходящих некоторое число n и взаимно простых с n.

Число натуральных чисел, не превосходящих n и, взаимно простых с n, называется функцией Эйлера и обозначается φ(n).

Понятие «модулярная арифметика» ввел немецкий ученый Гаусс. В этой арифметике мы интересуемся остатком от деления числа а на число n (n – натуральное число и n>1). Если таким остатком является число b, то можно записать: a ≡ b (mod n) или a ≡ b mod n. Такая формальная запись читается как «a сравнимо с b по модулю n». При целочисленном (в том числе и нулевом) результате деления числа а на число n справедливо a = b + k\*n. Иногда b называют вычетом по модулю n, а числа a и b называют сравнениями (по модулю n).

Модулярная арифметика так же коммутативна, ассоциативна и дистрибутивна, как и обычная арифметика. В силу этих свойств сравнения можно почленно складывать, вычитать, умножать, возводить в степень (аm mod n ≡ (а mod n) m, если a ≡ b mod n, то am ≡ bm (mod n). Указанные свойства позволяют упрощать сложность и время выполнения многих вычислений. Криптография использует множество вычислений по модулю n, потому что задачи типа вычисления дискретных логарифмов и квадратных корней очень трудны. Кроме того, с вычислениями по модулю удобнее работать, потому что они ограничивают диапазон всех промежуточных величин и результата. Обратные числа в модулярной арифметике. Традиционно: обратное к 7 равно 7-1=1/7, так как 7·(1/7) =1. В модулярной арифметике запись уравнения в виде aх ≡ 1 mod n предусматривает поиск таких значений х и k, которые удовлетворяют равенству aх = nk +1. Общая задача решения этого может быть сформулирована следующим образом: найти такое х, что 1≡ ах mod n. Это уравнение имеет единственное решение, если а и n – взаимно простые числа, в противном случае – решений нет. Это уравнение можно переписать в ином виде: а-1 ≡ х mod n. Нахождение чисел, обратных по модулю, легко реализуется с помощью расширенного алгоритма Евклида.

Малая теорема Ферма. Если n – простое число, а число а не кратно n, то справедливо: an ≡ 1 mod n. В соответствии с обобщением Эйлера приведенной теоремы, если НОД (а, n) =1, то справедливо: aφ(n) mod n ≡ 1. Последнее выражение можно переписать в следующем виде: а-1 mod n ≡ aφ(n)-1 mod n.

**Ход работы**

1. Найти все простые числа в интервале [2, 591]:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587

1. Подсчитать количество простых чисел в указанном интервале: N = 107
2. Сравнить это число с 591/ln(591).

591/ln(591) = 92,6 < N

1. Повторить п.1 для интервала [555, 591]. Сравнить полученные результаты с «ручными» вычислениями, используя «решето Эратосфена»

Простые числа в интервале [555,591]: 557, 563, 569, 571, 577, 587

1. Записать числа 555 и 591 в виде произведения простых множителей (форма записи – каноническая).

555 = 3 \* 5 \*37

591 = 3\*197

1. Проверить, является ли число 555591, простым - составное
2. Найти НОД (555, 591): НОД(555,591) = 3

**Описание** **приложения**

Приложение должно реализовывать следующие операции:

* вычислять НОД двух либо трех чисел;
* выполнять поиск простых чисел;
* находить обратно по модулю число.

Функция для нахождения простых чисел в диапазоне от 2 до N показана в листинге 3.

public static List<int> GetPrimes (int max)

{

if (max <= 0)

return null;

List<int> result = new List<int>();

for (int i = min; i <= max; i++)

if (IsSimple(i))

{

result.Add(i);

}

return result;

}

Листинг 1 – Функция для нахождения простых чисел

При это использовалась вспомогательная функция IsSimple, проверяющая, является ли число простым. Она представлена в листинге 2.

public static bool IsSimple(double x)

{

double sqrtX = Math.Sqrt(x);

for (int i = 2; i <= sqrtX; i++)

if (x % i == 0)

return false;

return true;

}

Листинг 2 – Функция для проверки простоты числа

Функция, реализующая подсчёт НОД для двух чисел показана в листинге 3.

public static int CountNOD(int a, int b)

{

while (a != b)

{

if (a > b)

{

int tmp = a;

a = b;

b = tmp;

}

b -= a;

}

return a;

}

Листинг 3 – Функция подсчёта наименьшего общего делителя для двух чисел

Вычисление наименьшего общего делителя для трёх чисел представляет собой последовательное вычисление НОД для первых двух чисел и вычисление НОД для результата и третьего числа. Реализация такой функции представлена в листинге 4.

public static int CountNOD(int a, int b, int c)

{

return CountNOD(CountNOD(a,b),c);

}

Листинг 4 – Функция подсчёта наименьшего общего делителя для трёх чисел

Функция для вычисления обратного числа по модулю представлена в листинге 5.

public static int Foo(int a, int m)

{

int x, y;

int g = GCD(a, m, out x, out y);

if (g != 1)

throw new ArgumentException();

return (x % m + m) % m;

}

Листинг 5 – Функция нахождения обратного числа по модулю

Функция, реализующая обратный алгоритм Евклида представлена в листинге 6.

public static int GCD(int a, int b, out int x, out int y)

{

if (a == 0)

{

x = 0;

y = 1;

return b;

}

int x1, y1;

int d = GCD(b % a, a, out x1, out y1);

x = y1 - (b / a) \* x1;

y = x1;

return d;

}

Листинг 6– Функция, реализующая расширенный алгоритм Евклида

**Вывод**

В результате выполнения данной лабораторной работы я приобрела навыки выполнения операций с числами для решения задач в области криптографии, разработала приложений для автоматизации этих операций, ознакомилась с особенностями реализации готового программного средства L\_PROST и особенностями выполнения с его помощью операций над простыми числами